

Fast matrix computations for functional additive models

Simon Barthelmé*

21st February 2014

Abstract

It is common in functional data analysis to look at a set of related functions: a set of learning curves, a set of brain signals, a set of spatial maps, etc. One way to express relatedness is through an additive model, whereby each individual function $g_i(x)$ is assumed to be a variation around some shared mean $f(x)$. Gaussian processes provide an elegant way of constructing such additive models, but suffer from computational difficulties arising from the matrix operations that need to be performed. Recently Heersink & Furrer have shown that functional additive model give rise to covariance matrices that have a specific form they called *quasi-Kronecker* (QK), whose inverses are relatively tractable. We show that under additional assumptions the two-level additive model leads to a class of matrices we call *restricted quasi-Kronecker* (rQK), which enjoy many interesting properties. In particular, we formulate matrix factorisations whose complexity scales only linearly in the number of functions in latent field, an enormous improvement over the cubic scaling of naïve approaches. We describe how to leverage the properties of rQK matrices for inference in Latent Gaussian Models.

1 Introduction

One of the most frequent scenarios in functional data analysis is that one has a group of related functions (for example, learning curves, growth curves, EEG traces, etc.) and the goal is to describe what these functions have in common and how they vary (Ramsay and Silverman, 2005; Behseta et al., 2005; Cheng et al., 2013). In that context functional additive models are very natural, and the simplest is the following two-level model:

$$g_i(x) = f(x) + d_i(x) \quad (1)$$

where functions $g_1(x), \dots, g_m(x)$ are m individual functions (e.g. learning curves for m different individuals), $f(x)$ is a mean function and $d_1(x) \dots d_n(x)$ describe individual differences.

The two-level model appears on its own, or as an essential building block in more complex models, which may include several hierarchical levels (as in functional ANOVA, Ramsay and Silverman, 2005; Kaufman and Sain, 2009; Sain et al., 2011), time shifts (Cheng et al., 2013; Kneip and Ramsay, 2008; Telesca and Inoue, 2008), or non-Gaussian observations (Behseta et al., 2005). Functional PCA (Ramsay and Silverman, 2005), which seeks to characterise the distribution of the difference functions d_i , is a closely related variant. In a Bayesian framework the two-level model can be expressed as a particular kind of Gaussian process prior (Kaufman and Sain, 2009). The goal of this paper is to show that the covariance matrices that arise in the two-level model have a form that lends itself to very efficient computation. We call these matrices *restricted quasi-Kronecker*, after Heersink and Furrer (2011).

The paper is structured as follows. We first give some background material on Gaussian processes and latent Gaussian models (Rue et al., 2009), and describe the particular covariance matrices that obtain in functional additive models. These matrices are *quasi-Kronecker* (QK) or *restricted quasi-Kronecker* (rQK), and in the next section we prove some theoretical results that follow from the *block-rotated* form of rQK matrices. In particular, rQK matrices have very efficient factorisations, and we detail a range of useful computations based on these factorisations. In the following section we apply our results to Gaussian data (joint smoothing), and show that marginal likelihoods and their derivatives are highly tractable. The other application we highlight concerns the modelling of spike trains, which leads to a latent Gaussian model with Poisson likelihood. We show that the Hessian matrices of the log-posterior in the two-level model are quasi-Kronecker, which makes the Laplace approximation and its derivative tractable. This leads to efficient approximate inference methods for large-scale functional additive models.

*University of Geneva, Department of Basic Neuroscience. simon.barthelme@unige.ch

1.1 Notation

The Kronecker product of \mathbf{A} and \mathbf{B} is denoted $\mathbf{A} \otimes \mathbf{B}$, the all-one vector of length n \mathbf{e}_n or \mathbf{e} if obvious from the context. Throughout m is used for the number of individual functions in the functional additive model, and n for the number of grid points.

In what follows we will use the following two properties of the Kronecker product and vec operators (Petersen and Pedersen, 2012):

$$(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) = (\mathbf{AC} \otimes \mathbf{BD}) \quad (2)$$

for compatible matrices, and

$$\text{vec}(\mathbf{AXB}) = (\mathbf{B}^t \otimes \mathbf{A}) \text{vec}(\mathbf{X}) \quad (3)$$

where the $\text{vec}(\mathbf{X})$ operator stacks the columns of \mathbf{X} vertically.

1.2 Gaussian process and latent Gaussian models

We give here a brief description of Gaussian processes (GPs), a much more detailed treatment is available in Rasmussen and Williams (2005). A GP with mean 0 and covariance function $k(x, x')$ is a distribution on the space of functions of some input space \mathcal{X} into \mathbb{R} , such that for every set of sampling points $\{x_1, \dots, x_n\}$, the sampled values $f(x_1), \dots, f(x_n)$ follow a multivariate Gaussian distribution

$$\begin{aligned} f(x_1), \dots, f(x_n) &\sim \mathcal{N}(0, \mathbf{K}) \\ \mathbf{K}_{ij} &= k(x_i, x_j) \end{aligned}$$

As a shorthand for the above, we will use the notation $f \sim GP(0, \kappa)$ in what follows. The covariance function usually expresses the idea that we expect the values of f at x_i and x_j to be close if x_i and x_j are themselves close. Often, covariance functions only depend on the distance between two points, so that $k(x_i, x_j) = \kappa(d(x_i, x_j))$, with $d(x_i, x_j)$ some distance function (usually Euclidean).

In most actual cases the covariance function is not known and involves two or more hyperparameters, which we will call θ . In machine learning the squared-exponential covariance function is especially popular:

$$\kappa_\theta(d) = \exp\left(-e^{-\theta_1} \frac{d^2}{2} + \theta_2\right) \quad (4)$$

where θ_1 is a (log) length-scale parameter and θ_2 controls the marginal variance of the process. For reasons of numerical stability we prefer to use a Matern 5/2 covariance function, which generates rougher functions but leads to covariance matrices that are better behaved. The Matern 5/2 covariance function has the following form:

$$\begin{aligned} \kappa_\theta(u) &= \left(1 + u + \frac{u^2}{3}\right) \exp(-u + \theta_2) \\ u &= \sqrt{5}d \times e^{-\theta_1} \end{aligned} \quad (5)$$

given in Rasmussen and Williams (2005), page 85. The parameters θ_1 and θ_2 play similar roles in this formulation, controlling length-scale and marginal variance.

Gaussian processes can be used to formulate priors for Latent Gaussian Models (Rue et al., 2009). The term ‘‘Latent Gaussian Model’’ describes a very large class of statistical models, one that encompasses for example all Generalised Linear Models (with Gaussian priors on the coefficients), Generalised Mixed Models, and Generalised Additive Models. The two main components are (a) a latent Gaussian field, $\mathbf{z} = [z_1 \dots z_n]$, with prior $\mathbf{z} \sim \mathcal{N}(0, \Sigma_\theta)$ and (b) independently distributed data $y_i \sim f(y|z_i)$, which depend on the corresponding value of the latent field. In regression models $f(y|z_i)$ is Gaussian, but other common distributions include a logit model (for binary y) and exponential-Poisson (as in section 3.2). The results we give here apply to all Latent Gaussian Models where the latent field has the structure of a two-level functional additive model, which we detail next.

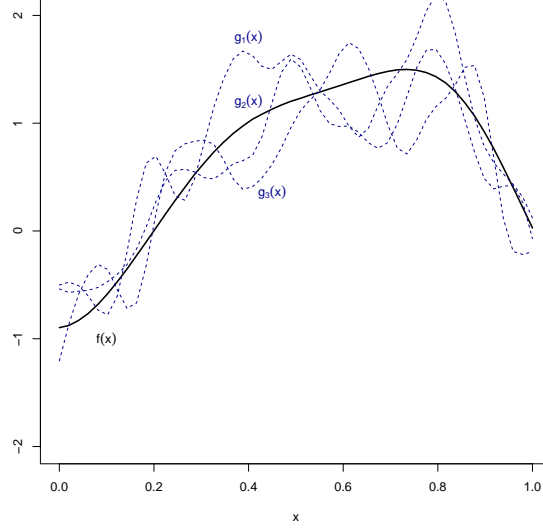


Figure 1: The functional additive model (eq. 6) produces sets of related functions. Here we show a latent mean function ($f(x)$), sampled from a Gaussian process. The individual functions g_1, g_2, g_3 are variations on f sampled from a Gaussian process with mean function $f(x)$.

1.3 Quasi-Kronecker structures in functional additive models

In a Gaussian process context the two-level functional additive model translates into the following model:

$$\begin{aligned} f &\sim GP(0, \kappa) \\ g_1 \dots g_m | f &\sim GP(f, \kappa') \end{aligned} \quad (6)$$

where individual g_i 's are conditionally independent given f (they are however marginally dependent).

Fig. 1 gives an example of a draw from such a model. Eq. 6 expresses the model as a prior over functions, but if we have a set of sampling locations $x_1 \dots x_n$ the Gaussian process assumption implies a multivariate Gaussian model for the latent field at the sample locations. We assume throughout that there are $n \times m$ observations, corresponding to m realisations of a latent process observed on a grid of size n , with grid points $x_1 \dots x_n$. The grid may be irregular (i.e. $x_{t+1} - x_t$ needs not equal a constant), but we need the grid to be constant over individual functions, since otherwise the covariance matrix does not have the requisite structure. Denote by \mathbf{g}_i the vector of sampled values from $g_i(x)$: $\mathbf{g}_i = [g_1(x_1) \dots g_n(x_n)]^t$. The latent field $G \in \mathbb{R}^{n \times m}$ is formed of the concatenation of $\mathbf{g}_1 \dots \mathbf{g}_m$, and we may think of it either as a $n \times m$ matrix \mathbf{G} or a single vector $\text{vec}(\mathbf{G})$. Following the LGM framework, we assume that the data are (possibly non-linear, non-Gaussian) observations based on the latent values, so that $y_{ij} \sim f(y|g_{ij})$.

Under a particular grid, sampled values from the mean function \mathbf{f} have distribution $\mathbf{f} \sim \mathcal{N}(0, \mathbf{K})$, and \mathbf{g}_i has conditional distribution $\mathbf{g}_i | \mathbf{f} \sim \mathcal{N}(\mathbf{f}, \mathbf{A})$. Draws from the latent field \mathbf{g} can therefore be obtained by the following transformation:

$$\begin{aligned} \mathbf{g} &= \begin{bmatrix} \mathbf{I} & \mathbf{I} & & \\ \mathbf{I} & & \mathbf{I} & \\ \vdots & & & \ddots \\ \mathbf{I} & & & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{f} \\ \mathbf{d}_1 \\ \vdots \\ \mathbf{d}_m \end{bmatrix} \\ &= \mathbf{M} \begin{bmatrix} \mathbf{f} \\ \mathbf{d}_1 \\ \vdots \\ \mathbf{d}_m \end{bmatrix} \end{aligned} \quad (7)$$

where $\mathbf{f} \sim \mathcal{N}(0, \mathbf{K})$ and $\mathbf{d}_i \sim \mathcal{N}(0, \mathbf{A})$. We can use eq. 7 to find the marginal distribution of \mathbf{g} (unconditional on \mathbf{f}). Using standard properties of Gaussian random variables we find:

$$\begin{aligned}
\mathbf{g} &\sim \mathcal{N}(\mathbf{0}, \Sigma) \\
\Sigma &= \mathbf{M} \begin{bmatrix} \mathbf{K} & & & \\ & \mathbf{A} & & \\ & & \dots & \\ & & & \mathbf{A} \end{bmatrix} \mathbf{M}^t \\
&= \begin{bmatrix} \mathbf{A} + \mathbf{K} & \mathbf{K} & \dots & \mathbf{K} \\ \mathbf{K} & \mathbf{A} + \mathbf{K} & \dots & \mathbf{K} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{K} & \mathbf{K} & \dots & \mathbf{A} + \mathbf{K} \end{bmatrix}
\end{aligned} \tag{8}$$

Σ is a dense matrix, whose dimensions ($nm \times nm$) would seem to preclude direct inference, since the costs of factorising such a matrix would be too high with numbers as low as 100 grid points and 20 observed functions. Heersink and Furrer (2011) have shown that the matrix inverse of Σ is actually surprising tractable, and in the following section we summarise and extend their results.

2 Some properties of restricted quasi-Kronecker matrices

Quasi-Kronecker (QK) matrices are introduced in Heersink and Furrer (2011) and have the following form:

$$\Sigma = \text{bdiag}(\mathbf{A}_1 \dots \mathbf{A}_m) + \mathbf{u}\mathbf{v}^t \otimes \mathbf{K} \tag{9}$$

We focus mostly on the restricted case (rQK), i.e. matrices of the form

$$\Sigma = \text{bdiag}(\mathbf{A} \dots \mathbf{A}) + \mathbf{e}\mathbf{e}^t \otimes \mathbf{K} = \mathbf{I}_m \otimes \mathbf{A} + \mathbf{e}\mathbf{e}^t \otimes \mathbf{K} = \text{rQK}(\mathbf{A}, \mathbf{K}) \tag{10}$$

which arise in the functional additive model described just above (eq. 8).

QK and especially rQK matrices have a number of properties that make them much more tractable than dense, unstructured matrices of the same size.

2.1 The general (unrestricted) case

The cost of matrix-vector products with QK matrices is $\mathcal{O}(n^2m)$, as compared to $\mathcal{O}(n^2m^2)$ in the general case. This follows directly from the definition (eq. 9), as we need only perform $\mathcal{O}(m)$ operations of complexity $\mathcal{O}(n^2)$.

Heersink and Furrer (2011) showed that the inverse and pseudo-inverse of QK matrices is also tractable. For the inverse the following formula applies:

$$\begin{aligned}
(\text{bdiag}(\mathbf{A}_1 \dots \mathbf{A}_m) + \mathbf{u}\mathbf{v}^t \otimes \mathbf{K})^{-1} &= \text{bdiag}(\mathbf{A}_i^{-1}) - \text{bdiag}(\mathbf{A}_i^{-1}) (\mathbf{u} \otimes \mathbf{L}_\mathbf{K}) \mathbf{P}^{-1} (\mathbf{v} \otimes \mathbf{L}_\mathbf{K})^t \text{bdiag}(\mathbf{A}_i^{-1}) \\
\mathbf{P} &= (\mathbf{I}_m + (\mathbf{v} \otimes \mathbf{L}_\mathbf{K})^t \text{bdiag}(\mathbf{A}_i^{-1}) (\mathbf{u} \otimes \mathbf{L}_\mathbf{K}))
\end{aligned} \tag{11}$$

where $\mathbf{K} = \mathbf{L}_\mathbf{K} \mathbf{L}_\mathbf{K}^t$. The result can be proved using the Sherman-Woodbury-Morrison formula, by writing $\mathbf{u}\mathbf{v}^t \otimes \mathbf{K} = (\mathbf{u} \otimes \mathbf{L}_\mathbf{K}) (\mathbf{v} \otimes \mathbf{L}_\mathbf{K})^t$, which follows from eq. (2). It implies that a $mn \times mn$ QK matrix can be inverted in $\mathcal{O}(n^3m + m^3)$ operations, as opposed to $\mathcal{O}(m^3n^3)$ in the general case. A similar formula holds for the determinant:

$$\det(\text{bdiag}(\mathbf{A}_1 \dots \mathbf{A}_m) + \mathbf{u}\mathbf{v}^t \otimes \mathbf{K}) = \det(\text{bdiag}(\mathbf{A}_1 \dots \mathbf{A}_m)) \det(\mathbf{P}) = \prod \det(\mathbf{A}_i) \det(\mathbf{P}) \tag{12}$$

where \mathbf{P} is as in equation 11. The result implies that determinants of QK matrices can be computed in $\mathcal{O}(n^3m + m^3)$ operations ($\mathcal{O}(m^3n^3)$ in the general case). We show in section 3.2 that it can be used to speed up Laplace approximations in latent Gaussian models (Rue et al., 2009).

2.2 The restricted case

rQK matrices have some additional properties not enjoyed by general QK matrices. For example, the product of two rQK matrices is another rQK matrix:

$$\text{rQK}(\mathbf{A}_1, \mathbf{K}_1) \text{rQK}(\mathbf{A}_2, \mathbf{K}_2) = \text{rQK}(\mathbf{A}_1 \mathbf{A}_2, \mathbf{A}_1 \mathbf{K}_2 + \mathbf{K}_1 \mathbf{A}_2 + m \mathbf{K}_1 \mathbf{K}_2) \quad (13)$$

We use this property below to compute the gradient of the marginal likelihood in Gaussian models (section 3.1.1). In addition, several other properties follow from the block-rotated form of rQK matrices: for example, the inverse of a rQK matrix is another rQK matrix, and the eigenvalue decomposition has a very structure. We prove these results next.

2.3 Block-rotated form of rQK matrices

In this section we first establish that rQK matrices can be *block-rotated* into a block-diagonal form. From this result the eigendecomposition follows immediately, and so does a Cholesky-based square root. These decompositions need not be formed explicitly, and we will see that their cost scales as $\mathcal{O}(n^2)$ in storage and $\mathcal{O}(mn^3)$ in time. The latter is linear in m , compared to cubic for naïve algorithms.

We begin by defining block rotations, which are straightforward extensions of block permutations. A block permutation can be written using the Kronecker product $\mathbf{P} \otimes \mathbf{I}_n$ of a $m \times m$ permutation matrix \mathbf{P} and the $n \times n$ identity matrix. Applied to a matrix \mathbf{M} , $(\mathbf{P} \otimes \mathbf{I}_n) \mathbf{M} (\mathbf{P}^t \otimes \mathbf{I}_n)$ will permute blocks of size $n \times n$. Block rotations are defined in a similar way, through the Kronecker product $\mathbf{R} = \mathbf{B} \otimes \mathbf{I}_n$ of a $m \times m$ rotation matrix \mathbf{B} and the $n \times n$ identity matrix. A block rotation matrix is an orthogonal matrix, since:

$$\begin{aligned} (\mathbf{B} \otimes \mathbf{I}_n)^t (\mathbf{B} \otimes \mathbf{I}_n) &= (\mathbf{B}^t \otimes \mathbf{I}_n) (\mathbf{B} \otimes \mathbf{I}_n) \\ &= (\mathbf{B}^t \mathbf{B} \otimes \mathbf{I}_n) \\ &= \mathbf{I}_{mn} \end{aligned}$$

where the second line follows from eq. (2).

Our main result is the following:

Theorem 1. *There exists an orthogonal matrix \mathbf{B} such that for all $n \times n$ matrices \mathbf{A}, \mathbf{K} , the matrix $\Sigma = \text{rQK}(\mathbf{A}, \mathbf{K})$ can be expressed as:*

$$\Sigma = (\mathbf{B} \otimes \mathbf{I}_n)^t \begin{pmatrix} (\mathbf{A} + m\mathbf{K}) & & & \\ & \mathbf{A} & & \\ & & \ddots & \\ & & & \mathbf{A} \end{pmatrix} (\mathbf{B} \otimes \mathbf{I}_n)$$

Since $\mathbf{R} = (\mathbf{B} \otimes \mathbf{I}_n)$ is an orthogonal matrix, and the inner matrix is block-diagonal, the inverse, eigendecomposition, and a Cholesky-based square root of Σ all follow easily.

Proof. We use the following ansatz: the $m \times m$ matrix \mathbf{B} is an orthogonal matrix whose first row is a scaled version of the all-ones vector

$$\mathbf{B} = \begin{bmatrix} \mathbf{e}^t / \sqrt{m} \\ \mathbf{L} \end{bmatrix}$$

and \mathbf{L} is chosen such that $\mathbf{B}^t \mathbf{B} = \mathbf{I}$, so that the rows of \mathbf{L} will be orthogonal to \mathbf{e} , and $\mathbf{B}\mathbf{e} = \begin{bmatrix} m/\sqrt{m} & 0 & \dots & 0 \end{bmatrix}$. Note that \mathbf{L} is not unique, but a matrix that verifies the condition can always be found by the Gram-Schmidt process (Golub and Van Loan, 1996) (although we will see below that a better option is available).

As noted above $\Sigma = (\mathbf{I}_m \otimes \mathbf{A}) + (\mathbf{e}\mathbf{e}^t \otimes \mathbf{K})$. We left-multiply by $\mathbf{R} = (\mathbf{B} \otimes \mathbf{I}_n)$ and right-multiply by \mathbf{R}^t :

$$\begin{aligned} \mathbf{R}\Sigma\mathbf{R}^t &= (\mathbf{B} \otimes \mathbf{I}_n) (\mathbf{I}_m \otimes \mathbf{A}) (\mathbf{B}^t \otimes \mathbf{I}_n) + (\mathbf{B} \otimes \mathbf{I}_n) (\mathbf{e}\mathbf{e}^t \otimes \mathbf{K}) (\mathbf{B}^t \otimes \mathbf{I}_n) \\ &= (\mathbf{B} \otimes \mathbf{A}) (\mathbf{B}^t \otimes \mathbf{I}_n) + (\mathbf{B}\mathbf{e}\mathbf{e}^t \otimes \mathbf{K}) (\mathbf{B}^t \otimes \mathbf{I}_n) \\ &= (\mathbf{B}\mathbf{B}^t \otimes \mathbf{A}) + (\mathbf{B}\mathbf{e}\mathbf{e}^t \mathbf{B}^t \otimes \mathbf{K}) \\ &= (\mathbf{I}_m \otimes \mathbf{A}) + \begin{pmatrix} m\mathbf{K} & 0 & \dots \\ 0 & 0 & \dots \\ \vdots & \vdots & \ddots \end{pmatrix} \end{aligned} \quad (14)$$

Left-multiplying by \mathbf{R}^t and right-multiplying by \mathbf{R} completes the proof. \square

2.4 Inverse, factorised form, and eigenvalue decomposition

A number of interesting properties follow directly from the block-rotated form given by Theorem 1.

Corollary 2. *The inverse of an rQK matrix is also an rQK matrix. Specifically,*

$$rQK(\mathbf{A}, \mathbf{K})^{-1} = rQK\left(\mathbf{A}^{-1}, \frac{1}{m} \left((\mathbf{A} + m\mathbf{K})^{-1} - \mathbf{A}^{-1}\right)\right)$$

Proof. This result can also be derived from the Furrer-Heersink formula (eq. 11) but follows naturally from Theorem 1. Assuming that $\mathbf{A} + m\mathbf{K}$ and \mathbf{K} have full rank, then the inverse of Σ exists and equals:

$$\Sigma^{-1} = (\mathbf{B} \otimes \mathbf{I}_n)^t \begin{pmatrix} (\mathbf{A} + m\mathbf{K})^{-1} & & & \\ & \mathbf{A}^{-1} & & \\ & & \ddots & \\ & & & \mathbf{A}^{-1} \end{pmatrix} (\mathbf{B} \otimes \mathbf{I}_n)$$

For Σ^{-1} to be an rQK matrix we need to find matrices \mathbf{A}' and \mathbf{K}' such that $\mathbf{A}' + m\mathbf{K}' = (\mathbf{A} + m\mathbf{K})^{-1}$, and $\mathbf{A}' = \mathbf{A}^{-1}$. This implies

$$\begin{aligned} \mathbf{A} + m\mathbf{K}' &= (\mathbf{A} + m\mathbf{K})^{-1} \\ \mathbf{K}' &= \frac{1}{m} \left((\mathbf{A} + m\mathbf{K})^{-1} - \mathbf{A}^{-1}\right) \end{aligned}$$

which requires the existence of $(\mathbf{A} + m\mathbf{K})^{-1}$ and \mathbf{A}^{-1} , both conditions being verified if Σ is invertible. \square

An important consequence of this result is that Hessian matrices in latent Gaussian models with prior covariance Σ turn out to be quasi-Kronecker (section 3.2), so that Laplace approximations can be computed in $\mathcal{O}(mn^3)$ time.

Corollary 3. *The eigenvalue decomposition of $\Sigma = rQK(\mathbf{A}, \mathbf{K})$ is given by*

$$\Sigma = \mathbf{R}^t \mathbf{N}^t \mathbf{D} \mathbf{N} \mathbf{R}$$

where \mathbf{R} is as in theorem 1, $\mathbf{N} = bdiag(ev(\mathbf{A} + m\mathbf{K}), ev(\mathbf{A}), \dots, ev(\mathbf{A}))$ and $\mathbf{D} = diag(\lambda(\mathbf{A} + m\mathbf{K}), \lambda(\mathbf{A}), \dots, \lambda(\mathbf{A}))$.

Proof. Since $\begin{pmatrix} (\mathbf{A} + m\mathbf{K}) & & & \\ & \mathbf{A} & & \\ & & \ddots & \\ & & & \mathbf{A} \end{pmatrix}$ is block-diagonal its eigendecomposition into $\mathbf{N}^t \mathbf{D} \mathbf{N}$ is straightforward. The

matrix $\mathbf{N} \mathbf{R}$ is orthogonal: $\mathbf{R}^t \mathbf{N}^t \mathbf{N} \mathbf{R} = \mathbf{I}$, and since $\Sigma = \mathbf{R}^t \mathbf{N}^t \mathbf{D} \mathbf{N} \mathbf{R}$ with diagonal \mathbf{D} we have its eigenvalue decomposition. \square

Corollary 4. *A set of square root factorisations of Σ is given by $\Sigma = \mathbf{G}^t \mathbf{G}$, with*

$$\mathbf{G} = \begin{pmatrix} \mathbf{U} & & & \\ & \mathbf{V} & & \\ & & \ddots & \\ & & & \mathbf{V} \end{pmatrix} \mathbf{R}$$

where $\mathbf{U}^t \mathbf{U} = \mathbf{A} + m\mathbf{K}$ and $\mathbf{V}^t \mathbf{V} = \mathbf{A}$.

This corollary is an entirely straightforward consequence of the theorem, but note that \mathbf{G} has special form, as the product of a block-diagonal matrix and a block-permutation matrix. In the next section we outline an algorithm that takes advantage of these properties to compute the factorisation with initial cost $\mathcal{O}(n^3)$ and further cost $\mathcal{O}(mn^2)$ per MVP.

2.5 A fast algorithm for square roots of Σ

Our fast algorithm is based on Corollary 4. Given a matrix square root $\Sigma = \mathbf{G}^t \mathbf{G}$, the typical operations used in statistical computation are the following:

- Correlating transform: given $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$ the vector $\mathbf{x} = \mathbf{G}^t \mathbf{z}$ is distributed according to $\mathbf{x} \sim \mathcal{N}(0, \Sigma)$. The correlating transform (MVP with \mathbf{G}^t) takes a IID vector and gives it the right correlation structure.
- “Whitening” transform: given $\mathbf{x} \sim \mathcal{N}(0, \Sigma)$ the vector $\mathbf{z} = \mathbf{G}^{-t} \mathbf{x}$ is distributed according to $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$. This is the inverse operation of the correlating transform and produces white noise from a correlated vector.
- Evaluation of $p(\mathbf{x}|\boldsymbol{\mu}, \Sigma) = (2\pi)^{-mn/2} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^t \Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu}) - \log \det \Sigma^{-1}\right)$. This can be done using the whitening transform but in addition the log-determinant of Σ is needed. We give a formula below.

Our algorithm is based on the square roots $\mathbf{U}^t \mathbf{U} = \mathbf{A} + m\mathbf{K}$ and $\mathbf{V}^t \mathbf{V} = \mathbf{A}$, which can be obtained from the Cholesky or eigendecompositions. In most cases the Cholesky version is faster but the eigendecomposition has advantages in certain contexts. We outline a generic algorithm here, the only practical difference being in solving the linear systems $\mathbf{U}^{-t} \mathbf{x}$ and $\mathbf{V}^{-t} \mathbf{x}$, which should be done via forward or back substitution if \mathbf{U} and \mathbf{V} are Cholesky factors (Golub and Van Loan, 1996).

2.5.1 Computing the correlating and whitening transforms

Corollary 4 tells us that the square root of Σ , \mathbf{G}^t equals:

$$\mathbf{G}^t = (\mathbf{B}^t \otimes \mathbf{I}_n) \begin{pmatrix} \mathbf{U}^t & & & \\ & \mathbf{V}^t & & \\ & & \ddots & \\ & & & \mathbf{V}^t \end{pmatrix} \quad (15)$$

We first show how to compute the correlating transform. Let $\mathbf{z} = \text{vec}([\mathbf{z}_1 \dots \mathbf{z}_m]) = \text{vec}(\mathbf{Z})$. Using eq. 3, we can rewrite $\mathbf{G}^t \mathbf{z}$ as:

$$\begin{aligned} \mathbf{G}^t \mathbf{z} &= (\mathbf{B}^t \otimes \mathbf{I}_n) \begin{pmatrix} \mathbf{U}^t & & & \\ & \mathbf{V}^t & & \\ & & \ddots & \\ & & & \mathbf{V}^t \end{pmatrix} \text{vec}(\mathbf{Z}) \\ &= (\mathbf{B}^t \otimes \mathbf{I}_n) \text{vec}([\mathbf{U} \mathbf{z}_1^t \quad \mathbf{V}^t \mathbf{z}_2 \quad \dots \quad \mathbf{V}^t \mathbf{z}_m]) \\ &= (\mathbf{B}^t \otimes \mathbf{I}_n) \text{vec}(\mathbf{Y}) \\ &= \text{vec}(\mathbf{YB}) \end{aligned}$$

Computing the correlating transform entails m MVP products with the square roots $\mathbf{U} \mathbf{z}_1^t \quad \mathbf{V}^t \mathbf{z}_2 \quad \dots \quad \mathbf{V}^t \mathbf{z}_m$ ($\mathcal{O}(n^2)$ in the general case) and m MVP products with \mathbf{B} . The latter m products can be done in $\mathcal{O}(mn)$ time, as we show below, meaning that the whole operation has total cost $\mathcal{O}(mn^2)$.

The whitening transform can be computed in a similar way:

$$\mathbf{G}^{-t} = \begin{pmatrix} \mathbf{U}^{-t} & & & \\ & \mathbf{V}^{-t} & & \\ & & \ddots & \\ & & & \mathbf{V}^{-t} \end{pmatrix} (\mathbf{B} \otimes \mathbf{I}_n) \quad (16)$$

so that for $\mathbf{x} = \text{vec}([\mathbf{x}_1 \dots \mathbf{x}_m]) = \text{vec}(\mathbf{X})$

$$\begin{aligned} \mathbf{G}^{-t} \mathbf{x} &= \begin{pmatrix} \mathbf{U}^{-t} & & & \\ & \mathbf{V}^{-t} & & \\ & & \ddots & \\ & & & \mathbf{V}^{-t} \end{pmatrix} (\mathbf{B} \otimes \mathbf{I}_n) \text{vec}(\mathbf{X}) \\ &= \text{bdiag}(\mathbf{U}^{-t}, \mathbf{V}^{-t}, \dots, \mathbf{V}^{-t}) \text{vec}(\mathbf{X}\mathbf{B}^t) \\ &= \text{bdiag}(\mathbf{U}^{-t}, \mathbf{V}^{-t}, \dots, \mathbf{V}^{-t}) \text{vec}(\mathbf{Y}) \\ &= \text{vec}([\mathbf{U}^{-t}\mathbf{y}_1 \quad \mathbf{V}^{-t}\mathbf{y}_2 \quad \dots \quad \mathbf{V}^{-t}\mathbf{y}_m]) \end{aligned}$$

and since matrix solves of the form $\mathbf{U}^{-t}\mathbf{y}$ and $\mathbf{V}^{-t}\mathbf{y}$ have $\mathcal{O}(n^2)$ cost for Cholesky or eigenfactors, the entire operation has the same cost as the correlating transform ($\mathcal{O}(mn^2)$).

2.5.2 Fast rotations

The whitening and correlating transforms above involve MVPs with a $m \times m$ orthogonal matrix $\mathbf{B} = \begin{bmatrix} \mathbf{e}^t/\sqrt{m} \\ \mathbf{L} \end{bmatrix}$, where \mathbf{L} is such that \mathbf{B} is orthonormal. Construction of \mathbf{L} via the Gram-Schmidt process would have initial cost $\mathcal{O}(m^3)$ and MVPs with \mathbf{B} $\mathcal{O}(m^2)$. There is however a particular choice for \mathbf{L} that brings these costs down to $\mathcal{O}(1)$ and $\mathcal{O}(m)$:

Proposition 5. *The matrix $\mathbf{L}^t = \begin{bmatrix} a\mathbf{1}_{m-1}^t \\ b + \mathbf{I}_{m-1} \end{bmatrix}$, with $a = \frac{1}{\sqrt{m}}$ and $b = -\left(\frac{1}{(m-1)}\left(1 + \frac{1}{\sqrt{m}}\right)\right)$ is an orthonormal basis for the set of zero-mean vectors $\mathcal{A}_m = \{\mathbf{u} \in \mathbb{R}^m \mid \sum u_i = 0\}$.*

The proof can be found in Appendix 5.1. With this choice of \mathbf{L} the matrix \mathbf{B} equals

$$\mathbf{B} = \begin{bmatrix} a & a & a & a & a \\ a & b+1 & b & b & b \\ a & b & b+1 & \dots & b \\ a & \vdots & \vdots & \ddots & b \\ a & b & b & \dots & b+1 \end{bmatrix} \quad (17)$$

and it is easy to check that $\mathbf{B}^t\mathbf{B} = \mathbf{I}$, $\mathbf{B}\mathbf{e} = [1 \ 0 \ \dots \ 0]$, and that in addition $\mathbf{B}^t = \mathbf{B}$, so that $\mathbf{X}\mathbf{B} = (\mathbf{B}^t\mathbf{X}^t)^t = (\mathbf{B}\mathbf{X})^t$. Due to the specific structure, a single MVP with \mathbf{B} has cost $\mathcal{O}(m)$ (which is the cost of multiplying the entries by a , by b and computing the sum).

2.5.3 Computing Gaussian densities and the determinant

In applications we will need to compute Gaussian densities of the following form: given $\mathbf{x} \sim \mathcal{N}(0, \Sigma)$

$$\begin{aligned} p(\mathbf{x}) &= (2\pi)^{-mn/2} \exp\left(-\frac{1}{2}\mathbf{x}^t\boldsymbol{\Sigma}^{-1}\mathbf{x} - \log \det \boldsymbol{\Sigma}^{-1}\right) \\ &= (2\pi)^{-mn/2} \exp\left(-\frac{1}{2}\mathbf{x}^t\mathbf{G}^{-t}\mathbf{G}^{-1}\mathbf{x} - \log \det \boldsymbol{\Sigma}^{-1}\right) \\ &= (2\pi)^{-mn/2} \exp\left(-\frac{1}{2}\mathbf{z}^t\mathbf{z} - \log \det \boldsymbol{\Sigma}^{-1}\right) \end{aligned} \quad (18)$$

where $\mathbf{z} = \mathbf{G}^{-1}\mathbf{x}$ is obtained via whitening. In addition, the log-determinant of $\boldsymbol{\Sigma}$ is needed. Similar matrices have the same determinant, so that $\det(\mathbf{R}\boldsymbol{\Sigma}\mathbf{R}^t) = \det(\boldsymbol{\Sigma})$. Further, for block-diagonal matrices, $\log \det(\text{bdiag}\mathbf{A}_i) = \sum \log \det \mathbf{A}_i$, which together with Theorem 1 implies that

$$\log \det \boldsymbol{\Sigma} = \log \det (\mathbf{A} + m\mathbf{K}) + (m-1) \log \det \mathbf{A} \quad (19)$$

These two log-determinants can be computed at $\mathcal{O}(n)$ cost from the Cholesky or eigendecompositions of $\mathbf{A} + m\mathbf{K}$ and \mathbf{A} , which are needed anyway for the computation of the square root of $\boldsymbol{\Sigma}$.

2.5.4 Summary

The cost of computing matrix square roots for quasi-Kronecker matrices is dominated by the initial cost of computing two $\mathcal{O}(n^3)$ decompositions. The resulting factors have $\mathcal{O}(n^2)$ storage cost and no further storage is required. Further operations (whitening, correlating, computing the determinant) come at much lower cost ($\mathcal{O}(m^2n)$ for whitening and correlating, $\mathcal{O}(n)$ for the determinant).

3 Applications

We first describe how to use our techniques in the linear-Gaussian setting. Under the assumption of Gaussian observations the marginal likelihood of the hyperparameters can be computed efficiently, and we show in addition how to compute the gradient of the marginal likelihood in $\mathcal{O}(n^3 + mn^2)$ cost. The resulting algorithm scales very well with m .

When the observations are not Gaussian, the marginal likelihood is unavailable in closed form. There are many ways to implement inference in this case, including variational inference (Opper and Archambeau, 2008), nested Laplace approximations (Rue et al., 2009), expectation propagation (Minka, 2001), various form of Markov Chain Monte Carlo (Neal, 1997; Rue and Held, 2005; Murray et al., 2010), or some combination of methods. We study in detail an example from neuroscience, smoothing of repeated spike train data, where the observations are Poisson distributed. We use a simple Laplace approximation, but our methods can be applied in a similar way to more sophisticated approximate inference or for MCMC sampling.

As stated in the introduction, we assume throughout that there are $n \times m$ observations, corresponding to m realisations of a latent process observed on a grid of size n , with grid points $x_1 \dots x_n$. The grid may be irregular but it is essential that it be constant *across realisations*, otherwise the covariance matrix does not have the requisite form. The observations may be represented as a $n \times m$ matrix \mathbf{Y} , or equivalently as a vector $\mathbf{y} = \text{vec}(\mathbf{Y})$.

3.1 Gaussian observations

In the Gaussian case the conditional distribution of the observations is assumed to be iid, with $\mathbf{y}_{ij} \sim \mathcal{N}(g_{ij}, \sigma^2)$. To simplify the notation we will sometimes absorb the noise variance σ^2 into the hyperparameters $\boldsymbol{\theta}$.

Depending on the scenario the goal of the analysis varies, and we might seek to estimate the latent functions $g_1(x), \dots, g_m(x)$, the latent mean function $f(x)$, or to make predictions for an unobserved function $g_{m+1}(x)$, etc. How we treat the hyperparameters may vary as well. In a “mixed modelling” spirit we might just want to compute their maximum likelihood estimate, or instead we may want to perform a full Bayesian analysis and produce samples from the posterior over hyperparameters, $p(\boldsymbol{\theta}|\mathbf{y})$.

In all of these cases the main quantities of interest are:

1. The (log-) marginal likelihood $\log p(\mathbf{y}|\boldsymbol{\theta}, \sigma^2)$ and optionally its derivatives with respect to the hyperparameters
2. The conditional regressions $p(\mathbf{f}|\mathbf{y}, \boldsymbol{\theta}, \sigma^2)$ and $p(\mathbf{g}|\mathbf{y}, \boldsymbol{\theta}, \sigma^2)$

Most of the other quantities are simply variants of the above and we will not go explicitly through all the calculations.

3.1.1 Computing the marginal likelihood and its derivatives

The marginal likelihood can be computed easily by noting that $\mathbf{y} = \mathbf{g} + \epsilon$, with $\epsilon \sim \mathcal{N}(0, \sigma^2 \mathbf{I})$, so that the marginal distribution of \mathbf{y} is

$$\mathbf{y} \sim \mathcal{N}(0, \boldsymbol{\Sigma} + \sigma^2 \mathbf{I})$$

and that if $\boldsymbol{\Sigma}$ has rQK structure, so does $\boldsymbol{\Sigma}' = \boldsymbol{\Sigma} + \sigma^2 \mathbf{I}$. Specifically, $\boldsymbol{\Sigma}' = \text{rQK}(\mathbf{A} + \sigma^2 \mathbf{I}, \mathbf{K}) = \text{rQK}(\mathbf{A}', \mathbf{K})$, and we may use formula (18) directly to compute $\log p(\mathbf{y}|\boldsymbol{\theta}, \sigma^2)$.

For maximum likelihood estimation of the hyperparameters (and for Hamiltonian Monte Carlo) it is useful to compute the gradient of $\log p(\mathbf{y}|\boldsymbol{\theta}, \sigma^2)$ with respect to the hyperparameters. To simplify notation we momentarily absorb the noise variance σ^2 into the hyperparameters $\boldsymbol{\theta}$ and note $\boldsymbol{\Sigma}$ the covariance matrix of \mathbf{y} .

An expression for the derivative with respect to a single hyperparameter θ_j is given by Rasmussen and Williams (2005) (eq. 5.9):

$$\frac{\partial}{\partial \theta_j} \log p(\mathbf{y}|\boldsymbol{\theta}) = \frac{1}{2} \left(\mathbf{y}^t \boldsymbol{\Sigma}^{-1} \left(\frac{\partial}{\partial \theta_j} \boldsymbol{\Sigma} \right) \boldsymbol{\Sigma}^{-1} \mathbf{y} - \text{tr} \left(\boldsymbol{\Sigma}^{-1} \left(\frac{\partial}{\partial \theta_j} \boldsymbol{\Sigma} \right) \right) \right) \quad (20)$$

It is straightforward to verify that if Σ is rQK, so is its derivative $\frac{\partial}{\partial \theta_j} \Sigma$, which implies that $\Sigma^{-1} \left(\frac{\partial}{\partial \theta_j} \Sigma \right)$ is also a rQK matrix (see section 2.3). Using that property along with theorem 1, some further algebra shows:

$$\text{tr} \left(\Sigma^{-1} \left(\frac{\partial}{\partial \theta_j} \Sigma \right) \right) = \text{tr} \left((\mathbf{A} + m\mathbf{K})^{-1} \frac{\partial}{\partial \theta_j} (\mathbf{A} + m\mathbf{K}) \right) + (m-1) \text{tr} \left(\mathbf{A}^{-1} \frac{\partial}{\partial \theta_j} (\mathbf{A}) \right) \quad (21)$$

Computing $\Sigma^{-1} \mathbf{y}$ can be done using the fast matrix square root, and $\left(\frac{\partial}{\partial \theta_j} \Sigma \right)$ is a rQK matrix, so that the dot product $\mathbf{y}^t \Sigma^{-1} \left(\frac{\partial}{\partial \theta_j} \Sigma \right) \Sigma^{-1} \mathbf{y}$ in eq. 20 is tractable as well (with complexity $\mathcal{O}(mn^2)$). Given that factorisations of $(\mathbf{A} + m\mathbf{K})$ and \mathbf{A} are needed to compute the marginal likelihood anyway, the extra cost in computing derivatives is relatively small (the factorisations have cost $\mathcal{O}(n^3)$, the rest is $\mathcal{O}(mn^2)$).

3.1.2 Conditional regressions

The conditional regressions are given by the posterior distributions $p(\mathbf{f}|\mathbf{y}, \boldsymbol{\theta}, \sigma^2)$ and $p(\mathbf{g}|\mathbf{y}, \boldsymbol{\theta}, \sigma^2)$. Both distributions are Gaussian, and their mean and covariance can be found through the usual Gaussian conditioning formulas:

$$\begin{aligned} E(\mathbf{f}|\mathbf{y}, \boldsymbol{\theta}, \sigma^2) &= \mathbf{Q}_f^{-1} (\mathbf{A} + \sigma^2 \mathbf{I})^{-1} \left(\sum_{i=1}^m \mathbf{y}_i \right) \\ \text{Cov}(\mathbf{f}|\mathbf{y}, \boldsymbol{\theta}, \sigma^2) &= \mathbf{Q}_f^{-1} \\ \mathbf{Q}_f &= \mathbf{K}^{-1} + m (\mathbf{A} + \sigma^2 \mathbf{I})^{-1} \end{aligned}$$

which involves $\mathcal{O}(n^3 + mn^2)$ computations, and:

$$\begin{aligned} E(\mathbf{g}|\mathbf{y}, \boldsymbol{\theta}, \sigma^2) &= \sigma^{-2} \mathbf{Q}_g^{-1} \mathbf{y} \\ \text{Cov}(\mathbf{g}|\mathbf{y}, \boldsymbol{\theta}, \sigma^2) &= \mathbf{Q}_g^{-1} \\ \mathbf{Q}_g &= \Sigma^{-1} + \sigma^{-2} \mathbf{I} \end{aligned}$$

where \mathbf{Q}_g is rQK and hence all computations are also $\mathcal{O}(n^3 + mn^2)$. Simultaneous confidence bands can be obtained from the diagonal elements of \mathbf{Q}_f^{-1} and \mathbf{Q}_g^{-1} .

3.1.3 Fast updates

“Fast” hyperparameter updates are often available in Gaussian process models (Neal, 2011), in the sense that one may quickly recompute the marginal likelihood $p(\mathbf{y}|\boldsymbol{\theta}')$ from the current value $p(\mathbf{y}|\boldsymbol{\theta})$. Usually a fast update means being able to skip a $\mathcal{O}(n^3)$ factorisation, and this may be done for rQK matrices as well. For example, the matrices \mathbf{A} , $\mathbf{A} + \tau \mathbf{I}$ and $\alpha \mathbf{A}$ (with $\alpha > 0$) all have the same eigenvectors, which means one can update certain hyperparameters without the need to recompute the square root of Σ from scratch. Although this strategy may result in speed-ups, it requires painstaking implementation and we have not explored it further (we note however that fast updates are also linked to fast gradient computations, since eq. 20 often simplifies). It may be worthwhile when used in combination with Neal’s (2011) strategy of separating fast and slow variables.

3.1.4 Results

Specialised vs. generic factorisation methods We first checked that implementing our specialised matrix factorisations is indeed worth the trouble (for reasonable problem sizes). Generic matrix factorisation techniques have been greatly optimised over the years and are often faster than an inefficient implementation of a specialised routine.

We implemented our matrix factorisation formulae in R and measured computing times for Gaussian densities $p(\mathbf{x}|\Sigma)$. The non-specialised routine needs to first form the matrix $\Sigma = \text{rQK}(\mathbf{A}, \mathbf{K})$ explicitly, and uses Cholesky factorisation to evaluate the density. Our specialised implementation comes under two variants, one based on Cholesky factors, and the other based on eigendecompositions, as explained in section 2.5. The algorithms were implemented in R and all benchmarks were executed on a standard desktop PC running R 3.0.1 under Ubuntu.

Figure 2 and 3 summarise the results. We varied m , the number of functions, for different values of n , the grid size. General-purpose factorisation scales with $\mathcal{O}(m^3 n^3)$, compared to our method, which scales with $\mathcal{O}(n^3 + mn^2)$. What we

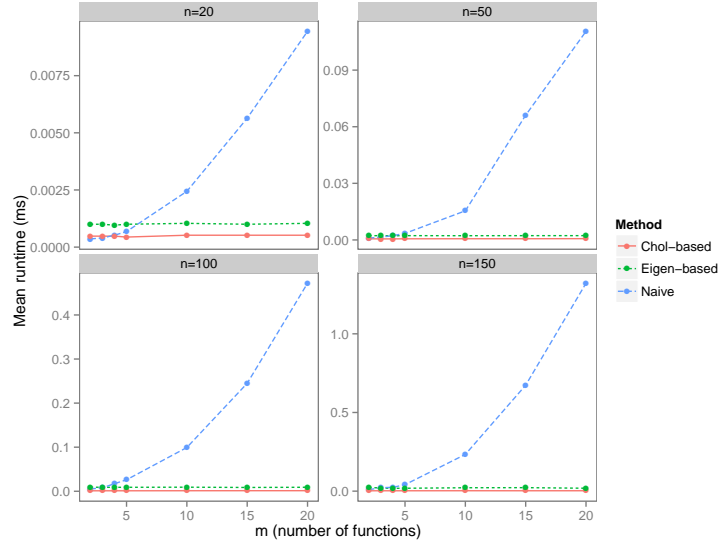


Figure 2: Relative efficiency of specialised versus generic methods for computing values of a multivariate Gaussian density. The blue line (“Naïve”) represents the runtime of a method based on a generic Cholesky factorisation of the covariance matrix. The other two lines represent runtimes of the specialised algorithms described here. One is based on an eigenvalue decomposition, the other on a Cholesky factor. Specialised methods are orders of magnitude faster for realistic problem sizes. . See also Fig 3.

expect, and what Fig. X verifies, is that our method should scale much better with m , as indeed it does: it is faster for all but the smallest problem sizes (e.g. with $n = 10$ and $m = 4$ our method is not worth the bother). With $n = 100$ we do better even with m as low as 2. What the results also underscore is the relative inefficiency of eigenvalue decompositions, relative to Cholesky factorisations. The eigenvalue decomposition remains of interest for regular grids, since in this case the Fourier decomposition can be used to factorise the covariance matrix in $\mathcal{O}(n \log n)$ operations (Paciorek, 2007), or if iterative methods are used in order to get an approximate factorisation (Saad, 2003).

Illustration in a joint smoothing problem We illustrate the application of our method in a joint smoothing problem. We generated data according to the following model:

$$\begin{aligned}
 f(x) &= \sin(12x) + \sin(24x) \\
 g_i(x) &= f(x) + w_{i1}\cos(6x) + w_{i2}\cos(3x) \\
 w_{ik} &\sim \mathcal{N}(0, 4) \\
 y_{ij} &= g_i(x_j) + \epsilon_{ij} \\
 \epsilon_{ij} &\sim \mathcal{N}(0, \sigma^2)
 \end{aligned}$$

meaning that the individual functions were smooth perturbations around a fixed mean function $f(x)$, as shown on Fig. 4.

We use a regular grid of points $x_i \in (0, 1)$.

We fitted a generic additive Gaussian process model, specifically:

$$\begin{aligned}
 f(x) &\sim GP(0, \kappa_K) \\
 g_i(x) &\sim GP(0, \kappa_A) \\
 y_{ij} &= g_i(x_j) + \epsilon_{ij}
 \end{aligned}$$

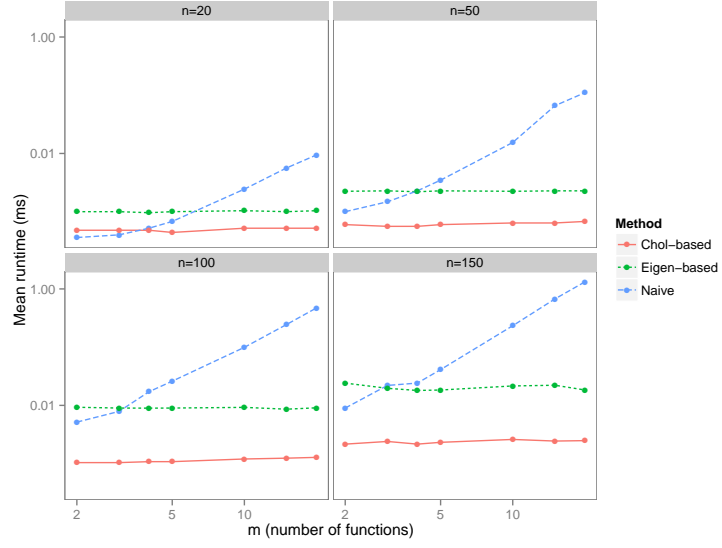


Figure 3: Same data as in fig. 2, replotted in log-log coordinates.

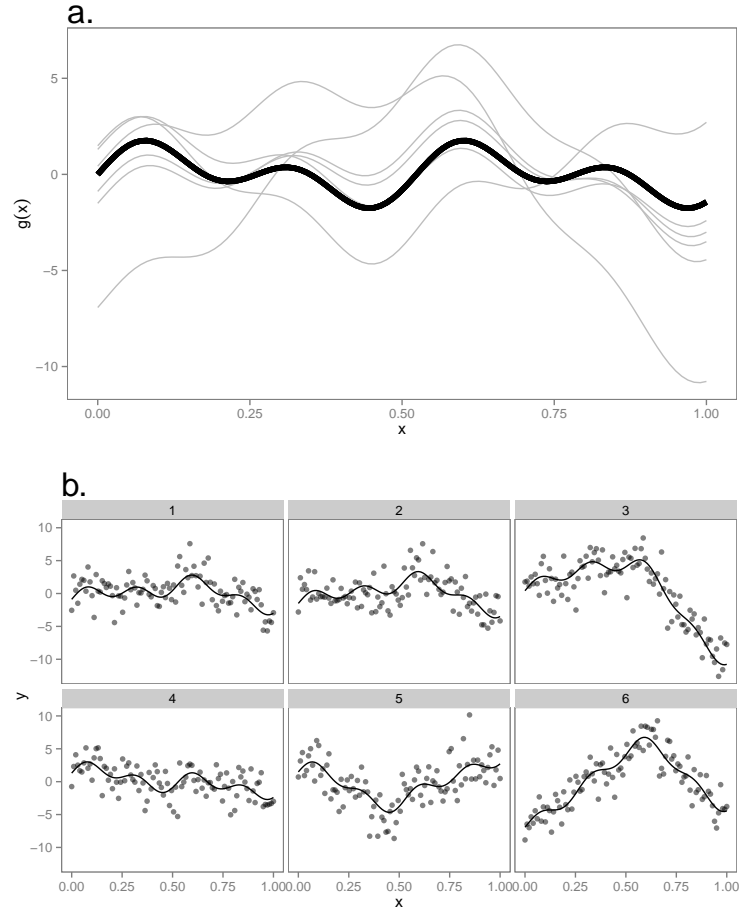


Figure 4: Data used in the simulations for the joint smoothing problem. **a.** The mean function $f(x)$ (thick lines) with 6 variants $g_1(x) \dots g_6(x)$ (thin lines) **b.** The data are noisy observations of the g_i 's (dots represent datapoints, lines correspond to the latent functions $g_1(x) \dots g_6(x)$). In this example $n = 100$ and the grid is regular.

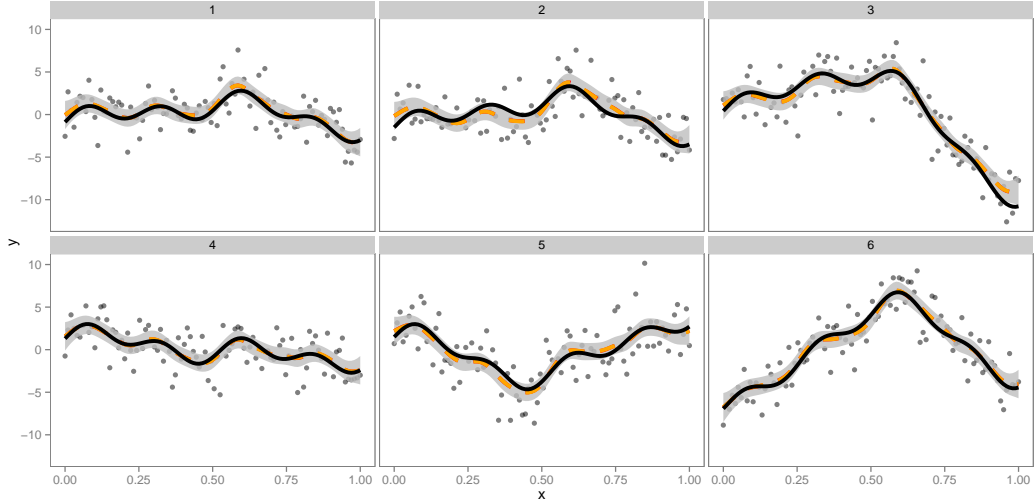


Figure 5: Fitted functions (posterior mean or MAP) vs. ground truth for the data shown in Fig. 4. Ground truth (actual values of $g_1(x) \dots g_6(x)$) is shown as a thick black lines, MAP and posterior mean fits are shown as orange and red dotted lines. MAP and posterior mean overlap nearly completely. The shaded region corresponds to a pointwise posterior confidence band from MCMC output.

The covariance functions κ_K and κ_A are Matern covariance functions, with fixed smoothness parameter $\nu = \frac{5}{2}$. The hyperparameters to estimate are the two length-scale parameters, the two Matern variance parameters, and the noise variance $\log \sigma^2$. We note θ the vector of hyperparameters. The most straightforward estimation strategy is to maximise $\log p(\mathbf{y}|\theta)$ (maximum likelihood, ML), or alternatively $\log p(\mathbf{y}|\theta) + \log p(\theta)$ (maximum a posteriori, MAP). MCMC can naturally also be used to sample from $p(\theta|\mathbf{y})$, and we compared both methods.

For MAP/ML we found that a quasi-Newton method (limited memory BFGS, Liu and Nocedal, 1989) works very well (we used the fast analytical gradient described above), converging typically in ~ 30 iterations. The standard Gaussian approximation of $p(\theta|\mathbf{y})$ at the mode θ^* can be computed by finite differences using the analytical gradient. For MCMC we used standard Metropolis-Hastings, with a proposal distribution corresponding to the Gaussian approximation of the posterior. When the data are informative enough the posterior is well-behaved and no further tuning is necessary, but problems can arise if e.g. the noise is very high and there are few measurements.

MAP inference is extremely fast (see fig. 7), and remains tractable with extremely large datasets: with a grid size of $n = 1,000$ and $m = 100$ latent functions, MAP inference takes a little over two minutes on our machine. MCMC is an order of magnitude slower but still rather tractable (it is likely that large speed-ups could be obtained using a modern Hamiltonian Monte Carlo method). Below we compare the results of MCMC and MAP inference for the data shown on fig. X.

Given a MAP estimate of the hyperparameters, inference for f and the g_i 's proceeds using the conditional posteriors $p(\mathbf{f}|\theta^*, \mathbf{y})$ and $p(\mathbf{g}|\theta^*, \mathbf{y})$. This tends to underestimate the uncertainty in the latent processes, but the underestimation is not dramatic in this example. The simultaneous confidence bands for MAP and full MCMC inference are compared on fig. 6 (appendix 5.3 explains how the confidence bands were estimated).

3.2 Non-Gaussian observations: application to spike count data

In this section we show how to apply our technique to LGMs with non-Gaussian likelihoods, using an example where the data are Poisson counts. The Poisson LGM is one of the most popular kinds, especially in spatial statistics applications (Illian et al., 2008; Barthelmé et al., 2013). Here we focus on a application to neuroscience, specifically spike count data.

Neurons communicate by sending electrical signals (action potentials, also called spikes) to one another. Each spike is a discrete event, and the occurrence of spikes may be recorded by placing an electrode in the vicinity of a cell. Neurons respond to stimuli by modulating how much they spike. An excited neuron increases its spike rate, an inhibited neuron decreases it. In experiments where spikes are recorded a typical task is to determine how the spike rate changes over time in response to a stimulus.

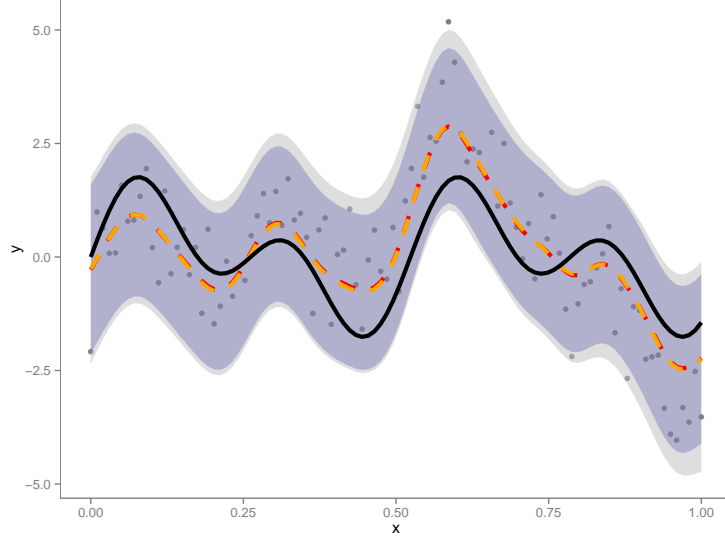


Figure 6: Posterior inference for the mean function $f(x)$ (same data as shown in fig. 4 and 5). Individual dots represent mean observation values at a given point (i.e. $\frac{1}{m} \sum_{i=1}^m y_{ij}$, which equals $f(x_j)$ in expectation). The true value of $f(x)$ is shown as a thick black line, MAP and posterior estimates are in orange and red respectively. Two 95% confidence bands are displayed: the lighter one is from MCMC inference, the darker one is an approximate interval obtained from the MAP estimate. The latter neglects posterior uncertainty over the hyperparameters and is therefore narrower; although the inference is not dramatic in this example, and both intervals contain the true value. Note that under the functional additive model the mean observations are not distributed i.i.d. around the mean function, making inference more difficult.

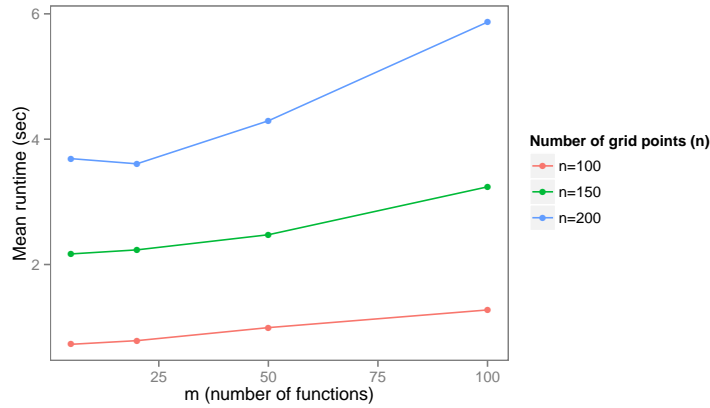


Figure 7: Runtime of MAP estimation of the hyperparameters (Gaussian case). We simulated data with different grid sizes (n) and different number of functions (m), and used numerical optimisation (L-BFGS-B) to estimate the hyperparameters. We show here average runtimes obtained over 30 repetitions with different random datasets. The method shows excellent scaling with m , the largest cost factor being again the $\mathcal{O}(n^3)$ matrix factorisations.

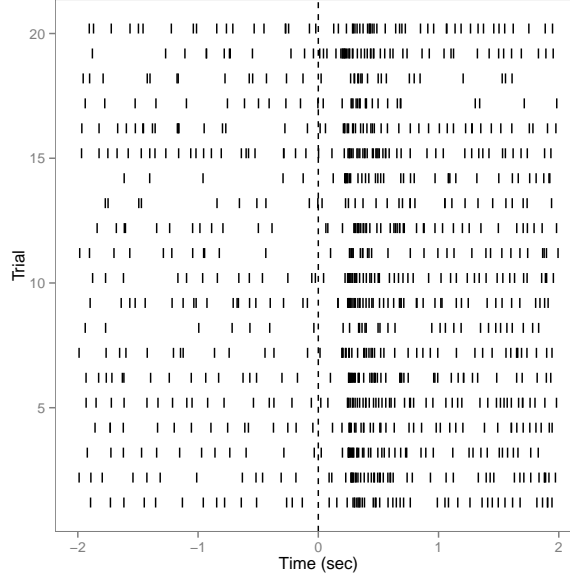


Figure 8: Spike train data used in the example: spiking responses of an antennal lobe neuron. Each vertical bar represents a spike, with successive trials stacked up vertically. Time $t = 0$ represents stimulus onset (stimulation with an odorant). The cell has an excitatory response: an increase in spike rate is visible around 200ms after stimulus onset. These data are available in the STAR package in R (Pouzat and Chaffiol, 2009, dataset e060817terpi, cell #1).

We use data provided by Pouzat and Chaffiol (2009), which consist in external recordings of a cell in the antennal lobe of a locust, an area that responds to olfactory stimulation. Part of the data is shown on Fig. 8. The animal was stimulated by an odorant (terpineol), causing a change in the spike rate of the cell. Stimulation was repeated over the course of 20 successive trials. We look at spiking activity occurring between -2 and +2 sec. relative to stimulus onset.

The data are spike counts: we simply count the number of spikes that occurred in a given time bin. Neurons are noisy, and statistical models for spike trains are generally variations on the Poisson model, which assumes that the spike count at time t follows a Poisson distribution with rate $\lambda(t)$ (Paninski et al., 2007). The goal is to infer the underlying rate $\lambda(t)$, which may vary spontaneously, drift from one trial to the next, and change in response to the stimulus.

We set up the following hierarchical model:

$$\begin{aligned} f(t) &\sim GP(0, \kappa) \\ g_i(t) &\sim GP(f(t), \kappa) \\ \lambda_i(t) &= \exp(g_i(t)) \\ y_{ij} &\sim \text{Poi}(\delta \lambda_i(t_j)) \end{aligned}$$

where δ is the time bin and $\delta \lambda_i(t_j) \approx \int_{t_j - \delta/2}^{t_j + \delta/2} \lambda_i(t) dt$ is the expected spike count in the j -th bin on the i -th trial.

3.3 Computing the Laplace approximation

Contrary to the Gaussian case, for generic LGMs the posterior marginals over the hyperparameters ($p(\mathbf{y}|\boldsymbol{\theta})$) cannot be easily computed. The Laplace approximation usually gives sensible results (Rue et al., 2009), although Expectation Propagation provides a superior if more expensive alternative (Nickisch and Rasmussen, 2008).

The Laplace approximation is given by:

$$\mathcal{L}(\boldsymbol{\theta}) = f(\mathbf{x}^*, \boldsymbol{\theta}) - \frac{1}{2} \log \det \mathbf{H}(\mathbf{x}^*, \boldsymbol{\theta}) \quad (22)$$

where

$$f(\mathbf{x}, \boldsymbol{\theta}) = \log p(\mathbf{y}|\mathbf{x}) + \log p(\mathbf{x}|\boldsymbol{\theta}) \quad (23)$$

is the unnormalised log-posterior evaluated at its conditional mode \mathbf{x}^* , and $\mathbf{H}(\mathbf{x}^*, \boldsymbol{\theta})$ is the Hessian of $f(\mathbf{x}, \boldsymbol{\theta})$ with respect to \mathbf{x} evaluated at \mathbf{x}^* , $\boldsymbol{\theta}$.

We therefore need to (a) find the conditional mode \mathbf{x}^* for a given value of $\boldsymbol{\theta}$ and (b) compute the log-determinant of the Hessian at the mode. The latter is possible because the Hessian turns out to be a QK matrix, and therefore its determinant can be computed in $\mathcal{O}(mn^3)$ using equation 12.

Proposition 6. *In LGMs the Hessian matrix of the log-posterior over the latent field is a quasi-Kronecker matrix.*

Proof. The second derivative of $f(\mathbf{x}, \boldsymbol{\theta})$ (eq. 23) with respect to \mathbf{x} is given by:

$$\begin{aligned}\nabla_{\mathbf{x}}^2 f(\mathbf{x}, \boldsymbol{\theta}) &= \nabla_{\mathbf{x}}^2 \log p(\mathbf{y}|\mathbf{x}) + \nabla_{\mathbf{x}}^2 \log p(\mathbf{x}|\boldsymbol{\theta}) \\ &= \mathbf{H}_l - \boldsymbol{\Sigma}^{-1}\end{aligned}$$

where the Hessian of the log-likelihood \mathbf{H}_l is diagonal (since each y_i depends only on x_i) and $\boldsymbol{\Sigma}^{-1}$ is rQK by Corollary 2. The sum of an rQK and a diagonal matrix is a QK matrix. \square

However, before we can do anything with the Hessian at the mode, we need to find the mode. Proposition 6 implies that one may use Newton’s method (Nocedal and Wright, 2006) to do so, since $\mathbf{H}^{-1}\nabla f$ can be computed using equation 11. However, at each Newton step we need to solve m linear systems of size $n \times n$, making it relatively expensive. We found that quasi-Newton methods (Nocedal and Wright, 2006), which do not use the exact Hessian, may be more efficient provided that one chooses a smart parametrisation.

Contrary to Newton’s method, which is invariant to linear transformations of the parameters, the performance of quasi-Newton methods depends partly on the conditioning of the Hessian. It is therefore worthwhile finding a transformation $\mathbf{z} = \mathbf{M}\mathbf{f}$ so that $\mathbf{M}\mathbf{H}\mathbf{M}^t$ is well-scaled. One option is to use the whitened parametrisation, i.e. $\mathbf{M} = \boldsymbol{\Sigma}^{-1/2} = \mathbf{G}^{-t}$ (eq. 16), in which case

$$\mathbf{M}\mathbf{H}\mathbf{M}^t = \mathbf{G}^{-t} (\boldsymbol{\Sigma}^{-1} + \mathbf{H}_l) \mathbf{G}^{-1} = \mathbf{I} + \mathbf{G}^{-t}\mathbf{H}_l\mathbf{G}^{-1} \quad (24)$$

Since one of the problems with Gaussian process priors is that $\boldsymbol{\Sigma}$ can have very poor conditioning, we might hope that the above transformation would help. We found that it does, but what is generally even more effective is to take into account the diagonal values of $\boldsymbol{\Sigma}^{-1/2}\mathbf{H}_l\boldsymbol{\Sigma}^{-1/2}$ as well (preconditioning the problem). Further discussion of the issue can be found in Appendix 5.4. The Quasi-Newton method we use is limited memory BFGS in the standard R implementation (*optim*).

3.4 Maximising the Laplace approximation

Once we have a way of approximating $p(\mathbf{y}|\boldsymbol{\theta})$, similar strategies apply in the general LGM case as do in the linear-Gaussian case. We may just require an approximate MAP estimate, or we may wish to approximately integrate out the uncertainty in $\boldsymbol{\theta}$ using INLA (Rue et al., 2009) or exact MCMC in a pseudo-marginal sampler (Filippone and Girolami, 2013). In any case the first step is to find the maximum of the Laplace approximation $\mathcal{L}(\boldsymbol{\theta})$. Most optimisation methods will work, but it helps if the gradient $\nabla \mathcal{L}(\boldsymbol{\theta})$ can be computed. It turns out that this is possible albeit rather more expensive than in the Gaussian case, and the derivation is given in Appendix 5.2.

3.5 Results

We first used the same point process models as in section 3.1 (Matern 5/2 for both \mathbf{K} and \mathbf{A} , with four hyperparameters). We used time bins of 2ms, giving a total of 200 time bins per spike train and 4,000 latent variables in total. We implemented the complete algorithm in R, and maximising the Laplace approximation takes a very reasonable 40 sec. The fitted intensity functions $\lambda_i(t) = \exp(g_i(t))$ are shown in fig. 9. There is a sharp increase in rate following stimulus onset, but little variability across trials. Stationary Gaussian processes such as the Matern process are not very well adapted to the estimation of functions that jump about a lot, and we feared that the ripples visible in the fits (e.g. prior to stimulus onset) could be artefactual. In other words, in order not to oversmooth around the jump, the model undersmooths in other regions.

As an alternative, we formulated a model that allows nonstationarity in the mean function $f(t)$, assuming:

$$\begin{aligned}f(t) &= a(t) + m(t)b(t) \\ m(t) &= \exp\left(-\frac{(t-t_0)^2}{s_t^2}\right)\end{aligned}$$

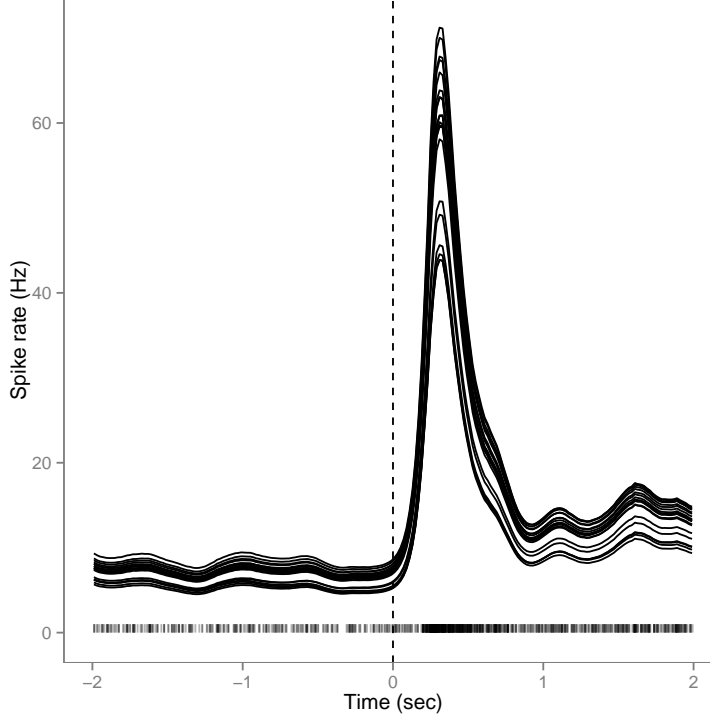


Figure 9: Fitted intensity functions for the spike train data of fig. 8, using stationary kernels. Each line represents the fitted function for a given trial. The strip plot at the bottom shows every spike in the dataset (over all trials). The “ripples” visible before stimulus onset and 1 sec. after onset are probably artefacts of the stationarity assumption, see next figure.

where $a(t)$ and $b(t)$ are two independent Matern processes, and $m(t)$ is a mask that limits the effect of $b(t)$ to the time around stimulus onset. The net effect is that extra variability is allowed around the time of the jump (see Bornn et al., 2012 for a more sophisticated approach to nonstationarity). We set $t_0 = 0.3$ and $s_t = 0.2$ by eye. Adding hyperparameters for $b(t)$ brings the total number of hyperparameters to 6, and fitting the nonstationary model takes about 2 minutes on our machine. The results, shown on fig. 10, indicate that pre-onset ripples are indeed most likely artefactual.

4 Conclusion

We have shown how restricted quasi-Kronecker matrices can be block rotated and factorised, and how this enables efficient inference in the two-level functional additive model. A similar closed-form factorisation for *generic* quasi-Kronecker matrices eludes us despite repeated attempts. That would certainly make an interesting topic for future work although perhaps not our own.

Although the algorithms we describe here scale very well with the number of functions in the dataset, they do not scale very well with grid size - the dreaded $\mathcal{O}(n^3)$ remains. For regular grids circulant embeddings (Fourier) approaches can be used (Paciorek, 2007), although unfortunately the Laplace approximation becomes hard to compute in that setting. Iterative solvers (Saad, 2003; Stein et al., 2013) could overcome the problem, but we leave that for future research.

Quasi-Kronecker and restricted quasi-Kronecker matrices display a number of appealing properties, and should probably join their block-diagonal, circulant, and Toeplitz peers among the set of computation-friendly positive definite matrices.

5 Appendix

5.1 An orthogonal basis for zero-mean vectors

We prove below proposition 5, which we restate:

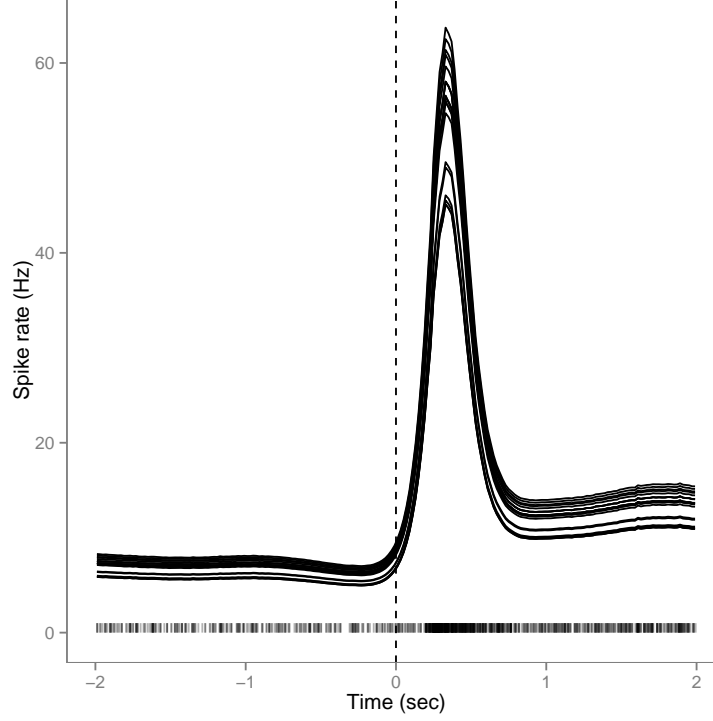


Figure 10: Same data as in fig. 9, fitted with a non-stationary kernel (see text). The ripples visible in fig. 9 have disappeared.

Proposition. The matrix $\mathbf{N} = \begin{bmatrix} a\mathbf{1}_{m-1}^t \\ b + \mathbf{I}_{m-1} \end{bmatrix}$, with $a = \frac{1}{\sqrt{m}}$ and $b = -\left(\frac{1}{(m-1)}\left(1 + \frac{1}{\sqrt{m}}\right)\right)$ is an orthonormal basis for the set of zero-mean vectors $\mathcal{A}_m = \{\mathbf{u} \in \mathbb{R}^m \mid \sum u_i = 0\}$.

Proof. For $\mathbf{N}_{m \times (m-1)}$ to be an orthogonal basis for \mathcal{A} , all that is required is that $\sum_{i=1}^m \mathbf{N}_{ij} = 0$ for all j and that $\mathbf{N}^t \mathbf{N} = \mathbf{I}$. To lighten the notation, define $r = m - 1$. The first condition requires

$$\begin{aligned} a + rb + 1 &= 0 \\ a &= -(1 + rb) \end{aligned} \tag{25}$$

and the second condition

$$\begin{aligned} \mathbf{N}^t \mathbf{N} &= \mathbf{I} \\ a^2 + rb^2 + 2b + \mathbf{I} &= \mathbf{I} \end{aligned}$$

implying that

$$a^2 + rb^2 + 2b = 0 \tag{26}$$

injecting 25 into 26, we get:

$$\begin{aligned} (1 + rb)^2 + rb^2 + 2b &= 0 \\ (r + r^2)b^2 + (2 + 2r)b + 1 &= 0 \end{aligned}$$

This is a 2nd order polynomial in b , and it has real roots if:

$$\begin{aligned} (2 + 2r)^2 - 4(r + r^2) &> 0 \\ 4 + 4r &> 0 \end{aligned}$$

which is true since $r = m - 1$ is non-negative.

Solving the quadratic equations yields $b = -\left(\frac{1}{r} + \frac{1}{r\sqrt{m}}\right)$ and substituting into 25 yields $a = \frac{1}{\sqrt{m}}$.

We can therefore always find values a, b such that $\mathbf{N} = \begin{bmatrix} a\mathbf{1}_{m-1}^t \\ b + \mathbf{I}_{m-1} \end{bmatrix}$ is an orthogonal basis for the linear subset of zero-mean vectors. Moreover, we can show that the columns of \mathbf{N} have unit norm:

$$\begin{aligned} \sum_i \mathbf{N}_{ij}^2 &= a^2 + (b+1)^2 + (m-2)b^2 \\ &= a^2 + (m-1)b^2 + 2b + 1 \\ &= 1 \end{aligned}$$

where the last line is from 26. The result implies that we can compute an orthonormal basis \mathbf{N} for \mathcal{A} such that matrix-vector products with \mathbf{N} are $\mathcal{O}(m)$. \square

5.2 Derivative of the Laplace approximation

Using the implicit function theorem an analytical expression for the gradient of the Laplace approximation can be found (see Rasmussen and Williams, 2005, for a similar derivation). The Laplace approximation is given by:

$$\mathcal{L}(\boldsymbol{\theta}) = f(\mathbf{x}^*, \boldsymbol{\theta}) - \frac{1}{2} \log \det \mathbf{H}(\mathbf{x}^*, \boldsymbol{\theta}) \quad (27)$$

where

$$f(\mathbf{x}, \boldsymbol{\theta}) = \log p(\mathbf{y}|\mathbf{x}) + \log p(\mathbf{x}|\boldsymbol{\theta}) \quad (28)$$

is the unnormalised log-posterior evaluated at its conditional mode \mathbf{x}^* , and $\mathbf{H}(\mathbf{x}^*, \boldsymbol{\theta})$ is the Hessian of $f(\mathbf{x}, \boldsymbol{\theta})$ with respect to \mathbf{x} evaluated at $\mathbf{x}^*, \boldsymbol{\theta}$. Since \mathbf{x}^* is a maximum it satisfies the gradient equation:

$$\begin{aligned} F(\mathbf{x}, \boldsymbol{\theta}) &= 0 \\ F(\mathbf{x}, \boldsymbol{\theta}) &= \nabla \log p(\mathbf{y}|\mathbf{x}) + \nabla \log p(\mathbf{x}|\boldsymbol{\theta}) \end{aligned}$$

Assuming that $f(\mathbf{x}, \boldsymbol{\theta})$ is twice-differentiable and concave in \mathbf{x} , we can define an implicit function $\mathbf{x}^*(\boldsymbol{\theta})$ such that $F(\mathbf{x}^*(\boldsymbol{\theta}), \boldsymbol{\theta}) = 0$ for all $\boldsymbol{\theta}$, with derivative

$$\begin{aligned} \frac{\partial}{\partial \theta_j} \mathbf{x}^* &= -\left(\frac{\partial}{\partial \mathbf{x}} F(\mathbf{x}, \boldsymbol{\theta})\right)^{-1} \left(\frac{\partial}{\partial \boldsymbol{\theta}} F(\mathbf{x}, \boldsymbol{\theta})\right) \\ &= -(\mathbf{H}(\mathbf{x}^*(\boldsymbol{\theta}), \boldsymbol{\theta}))^{-1} \left(\boldsymbol{\Sigma}^{-1} \left(\frac{\partial}{\partial \theta_j} \boldsymbol{\Sigma}\right) \boldsymbol{\Sigma}^{-1} \mathbf{x}(\boldsymbol{\theta})\right) \end{aligned} \quad (29)$$

To simplify the notation we will note \mathbf{G} the matrix $\boldsymbol{\Sigma}^{-1} \left(\frac{\partial}{\partial \theta_j} \boldsymbol{\Sigma}\right) \boldsymbol{\Sigma}^{-1}$. Note that the same \mathbf{G} matrix appears in the gradient of the Gaussian likelihood with respect to the hyperparameters (eq. 20). To compute the gradient of $\mathcal{L}(\boldsymbol{\theta})$, we need the derivatives of $f(\mathbf{x}^*(\boldsymbol{\theta}), \boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$:

$$\frac{\partial}{\partial \boldsymbol{\theta}} f = \left(\frac{\partial}{\partial \mathbf{x}^*} f(\mathbf{x}^*, \boldsymbol{\theta})\right) \left(\frac{\partial}{\partial \theta_j} \mathbf{x}^*\right) + \frac{\partial}{\partial \theta_j} f(\mathbf{x}^*(\boldsymbol{\theta}), \boldsymbol{\theta})$$

where the first part is 0 since the gradient of f is 0 at $\mathbf{x}^*(\boldsymbol{\theta})$, and

$$\frac{\partial}{\partial \theta_j} f(\mathbf{x}, \boldsymbol{\theta}) = \frac{\partial}{\partial \theta_j} (\log p(\mathbf{y}|\mathbf{x}) + \log p(\mathbf{x}|\boldsymbol{\theta})) = \frac{\partial}{\partial \theta_j} \log p(\mathbf{x}|\boldsymbol{\theta})$$

This is the gradient of a log-Gaussian density, and we can therefore reuse formula 20 for that part.

The gradient of $\log \det \mathbf{H}(\mathbf{x}^*, \boldsymbol{\theta})$ is also needed and is sadly more troublesome. A formula for the derivative of the log-det function is given in Petersen and Pedersen (2012):

$$\frac{\partial}{\partial \theta_j} \log \det \mathbf{H} = \text{Tr} \left(\mathbf{H}^{-1} \frac{\partial}{\partial \theta_j} \mathbf{H} \right) \quad (30)$$

The Hessian at the mode is the sum of the Hessian of $\log p(\mathbf{y}|\mathbf{x}^*)$ (which depends on θ through the implicit dependency of \mathbf{x}^* on θ), and the Hessian of $\log p(\mathbf{x}|\theta)$, which equals the inverse covariance matrix Σ^{-1} . Some additional algebra yields:

$$\frac{\partial}{\partial \theta_j} \mathbf{H} = \left(\text{diag} \left(\left(\frac{\partial}{\partial \theta_j} \mathbf{x}_i^* \right) \frac{\partial^3}{\partial^3 x_i} \log p(\mathbf{y}|\mathbf{x}) \right) - \Sigma^{-1} \left(\frac{\partial}{\partial \theta_j} \Sigma \right) \Sigma^{-1} \right) \quad (31)$$

$$= (\mathbf{D} - \mathbf{G}) \quad (32)$$

where we have assumed that the Hessian of the log-likelihood is diagonal. Accordingly \mathbf{H}^{-1} is quasi-Kronecker, and so is $\frac{\partial}{\partial \theta_j} \mathbf{H}$ (since it equals the sum of a rQK matrix $-\mathbf{G}$ and a diagonal perturbation \mathbf{D}). Computing the trace in eq. (30) is therefore tractable if slightly painful, and can be done by plugging in eq. 11 and summing the diagonal elements.

5.3 Computing simultaneous confidence bands

In this section we outline a simple method for obtaining a Rao-Blackwellised confidence band from posterior samples of $p(\theta|\mathbf{y})$. A simultaneous confidence band around a latent function $f(x)$ is a vector function $c_\alpha(x) = \begin{pmatrix} h_\alpha(x) \\ l_\alpha(x) \end{pmatrix}$, such that, for all x :

$$\begin{aligned} p(f(x) > h_\alpha(x) | \mathbf{y}) &< 1 - \alpha \\ p(f(x) \leq l_\alpha(x) | \mathbf{y}) &< \alpha \end{aligned}$$

In latent Gaussian models the posterior over latent functions is a mixture over conditional posteriors:

$$p(f(x) | \mathbf{y}) = \int p(f|\mathbf{y}, \theta) p(\theta|\mathbf{y}) d\theta \quad (33)$$

The conditional posteriors $p(f(t) | \mathbf{y}, \theta)$ are either Gaussian or approximated by a Gaussian. If we have obtained samples from $p(\theta|\mathbf{y})$ (as in section 3.1) we can approximate (33) using a mixture of Gaussians, which has an analytic cumulative density function. The c.d.f. can be inverted numerically to obtain values $h_\alpha(t)$ and $l_\alpha(t)$.

5.4 Preconditioning for quasi-Newton optimisation

In the whitened parametrisation the Hessian matrix has the following form (eq 24)

$$\mathbf{H}_z(\mathbf{z}) = \mathbf{I} + \mathbf{G}^{-t} \mathbf{H}_l (\mathbf{G}^t \mathbf{z}) \mathbf{G}^{-1}$$

Although the whitened parametrisation gives ideal conditioning for the prior term, the conditioning with respect to the likelihood term (\mathbf{H}_l) may be degraded. Preconditioning the problem can help tremendously: given a guess \mathbf{z}_0 , one precomputes the diagonal values of $\mathbf{d}_0 = \text{diag}(\mathbf{I} + \mathbf{G}^{-t} \mathbf{H}_l (\mathbf{G}^t \mathbf{z}_0) \mathbf{G}^{-t})$ and uses $\text{diag}(\mathbf{d}_0)$ as a preconditioner. To compute \mathbf{d}_0 the following identity is useful:

$$\text{diag}(\mathbf{U} \text{diag}(\mathbf{v}) \mathbf{U}^t) = (\mathbf{U} \odot \mathbf{U}) \mathbf{v}$$

where \odot is the element-wise product. In LGMs \mathbf{H}_l is diagonal and so:

$$\mathbf{d}_0 = \mathbf{1} + (\mathbf{G}^{-t} \odot \mathbf{G}^{-t}) \mathbf{G}^t \mathbf{z}_0 \quad (34)$$

We inject 16 into the above and some rather tedious algebra shows that \mathbf{d}_0 can be computed in the following way. Form the matrix \mathbf{X}_0 by stacking successive subsets of length n from $\mathbf{G}^t \mathbf{z}_0$. Rotate using the matrix \mathbf{B} (section 2.5.2) to form another matrix:

$$\mathbf{Y} = \mathbf{X}_0 \mathbf{B}$$

then compute \mathbf{d}_0 from:

$$\mathbf{d}_0 = \text{vec} \left(\begin{bmatrix} (\mathbf{U} \odot \mathbf{U}) \mathbf{Y}_{:,1} & (\mathbf{V} \odot \mathbf{V}) \mathbf{Y}_{:,2:m} \end{bmatrix} \right)$$

where we have used Matlab notation subsets of columns of \mathbf{Y} , and \mathbf{U} and \mathbf{V} are as in Corollary 4.

Acknowledgements

The author wishes to thank Reinhard Furrer for pointing him to Heersink and Furrer (2011), and Alexandre Pouget for support.

References

- Barthelmé, S., Trukenbrod, H., Engbert, R., and Wichmann, F. (2013). Modeling fixation locations using spatial point processes. *Journal of vision*, 13(12).
- Behseta, S., Kass, R. E., and Wallstrom, G. L. (2005). Hierarchical models for assessing variability among functions. *Biometrika*, 92(2):419–434.
- Bornn, L., Shaddick, G., and Zidek, J. V. (2012). Modeling Nonstationary Processes Through Dimension Expansion. *Journal of the American Statistical Association*, 107(497):281–289.
- Cheng, W., Dryden, I. L., and Huang, X. (2013). Bayesian registration of functions and curves.
- Filippone, M. and Girolami, M. (2013). Exact-Approximate bayesian inference for gaussian processes.
- Golub, G. H. and Van Loan, C. F. (1996). *Matrix Computations (3rd Edition)*. Johns Hopkins University Press, 3rd edition.
- Heersink, D. K. and Furrer, R. (2011). On moore-penrose inverses of quasi-kronecker structured matrices. *Linear Algebra and its Applications*.
- Illian, J., Penttinen, A., Stoyan, H., and Stoyan, D. (2008). *Statistical Analysis and Modelling of Spatial Point Patterns (Statistics in Practice)*. Wiley-Interscience, 1 edition.
- Kaufman, C. and Sain, S. (2009). Bayesian functional ANOVA modeling using Gaussian process prior distributions. *Bayesian Analysis*, 5:123–150.
- Kneip, A. and Ramsay, J. O. (2008). Combining Registration and Fitting for Functional Models. *Journal of the American Statistical Association*, 103(483):1155–1165.
- Liu, D. and Nocedal, J. (1989). On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1-3):503–528.
- Minka, T. P. (2001). Expectation propagation for approximate bayesian inference. In *UAI '01: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, pages 362–369, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.
- Murray, I., Adams, R. P., and MacKay, D. J. (2010). Elliptical slice sampling. *JMLR: W&CP*, 9:541–548.
- Neal, R. M. (1997). Monte Carlo Implementation of Gaussian Process Models for Bayesian Regression and Classification.
- Neal, R. M. (2011). MCMC using ensembles of states for problems with fast and slow variables such as gaussian process regression.
- Nickisch, H. and Rasmussen, C. E. (2008). Approximations for Binary Gaussian Process Classification. *Journal of Machine Learning Research*, 9:2035–2078.
- Nocedal, J. and Wright, S. (2006). *Numerical Optimization (Springer Series in Operations Research and Financial Engineering)*. Springer, 2nd edition.
- Opper, M. and Archambeau, C. (2008). The variational gaussian approximation revisited. *Neural Computation*, 21(3):786–792.
- Paciorek, C. J. (2007). Bayesian Smoothing with Gaussian Processes Using Fourier Basis Functions in the spectralGP Package. *Journal of statistical software*, 19(2).
- Paninski, L., Pillow, J., and Lewi, J. (2007). Statistical models for neural encoding, decoding, and optimal stimulus design. *Progress in brain research*, 165:493–507.

- Petersen, K. B. and Pedersen, M. S. (2012). The matrix cookbook. Version 20121115.
- Pouzat, C. and Chaffiol, A. (2009). Automatic spike train analysis and report generation. an implementation with r, R2HTML and STAR. *Journal of Neuroscience Methods*, 181(1):119–144.
- Ramsay, J. and Silverman, B. W. (2005). *Functional Data Analysis (Springer Series in Statistics)*. Springer, 2nd edition.
- Rasmussen, C. E. and Williams, C. K. I. (2005). *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning series)*. The MIT Press.
- Rue, H. and Held, L. (2005). *Gaussian Markov Random Fields: Theory and Applications (Chapman & Hall/CRC Monographs on Statistics & Applied Probability)*. Chapman and Hall/CRC, 1 edition.
- Rue, H., Martino, S., and Chopin, N. (2009). Approximate bayesian inference for latent gaussian models by using integrated nested laplace approximations. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(2):319–392.
- Saad, Y. (2003). *Iterative Methods for Sparse Linear Systems, Second Edition*. Society for Industrial and Applied Mathematics, 2 edition.
- Sain, S. R., Nychka, D., and Mearns, L. (2011). Functional ANOVA and regional climate experiments: a statistical analysis of dynamic downscaling. *Environmetrics*, 22(6):700–711.
- Stein, M. L., Chen, J., and Anitescu, M. (2013). Stochastic approximation of score functions for Gaussian processes. *The Annals of Applied Statistics*, 7(2):1162–1191.
- Telesca, D. and Inoue, L. Y. T. (2008). Bayesian Hierarchical Curve Registration. *Journal of the American Statistical Association*, 103(481):328–339.